# Project 4

## Project Name: Sampling-Based Navigation

## Name of the students: Shaurya Panthri and Prajval Vaskar

We implemented the probabilistic roadmap method to answer motion planning queries for a 2D robot moving in an environment filled with static obstacles. In the basic requirements, we assumed that the robot is a translating disk.

Two phases of PRM:

1. Construction phase:

1. Sampling strategy:

There are two strategy that we used.

1. Construction Phase:
   1. Uniform sampling with Gaussian noise provided the best optimized results: We created samples using the numpy linspace command in both x and y direction from the lower limit to upper limit of the maze with tolerance equal to the radius of the robot. After several iterations we achieved the optimized results when the number of sample sets was 35. In order to generate random noise across the points we used numpy random uniform within a range of -0.8 to 0.8 and randomly chose these values using numpy random choice and added them to the points in the x and y axes. Thus, random vertices were generated using these two functions and a sampling strategy with tolerance to keep the vertices within the bounds.
   2. In order to select the vertices and avoid loopy paths we initially used a function to with flags to avoid connections more than 2 but it did not work as the edges were not present on samples that were created in portion of the map where the vertices were already connected with edges. We even tried to eliminate these extra vertices but that messed up the graph and ran few errors. Therefore, after using DFS (Depth First Search) we finally moved on to BFS (Breadth First Search) that was a recursive process to ensure that every vertex is connected via edges with its nearby children. This was the approach to make sure that locally the path is complete to implement the A star search. We reduced the computation cost by selecting a distance maximum of 5 for the edges and therefore we did not have to check for the obstacles. The portion is commented in the code.
2. Query Phase:
   1. First of all, we added the start and the end nodes to the Roadmap using the add Vertex. After that we calculated the nearest neighbors for the start and the end vertices of the sample created earlier. The edges within the minimum distance from the start and end nodes for default or the random query are hence joined with the vertices (and the positions of these using the argmin function). We used the ordered set and priority que to implement a star. The while loop iterates until the open set is empty that is when the Robot has reached the goal. At every step the control checks if the node is already visited or not and then calculates the g and h that is using the Euclidean distance calculation. Thus, f is calculated and added to the open set so that the one with least cost can be extracted and put inside the closed set that gives the path of the robot.

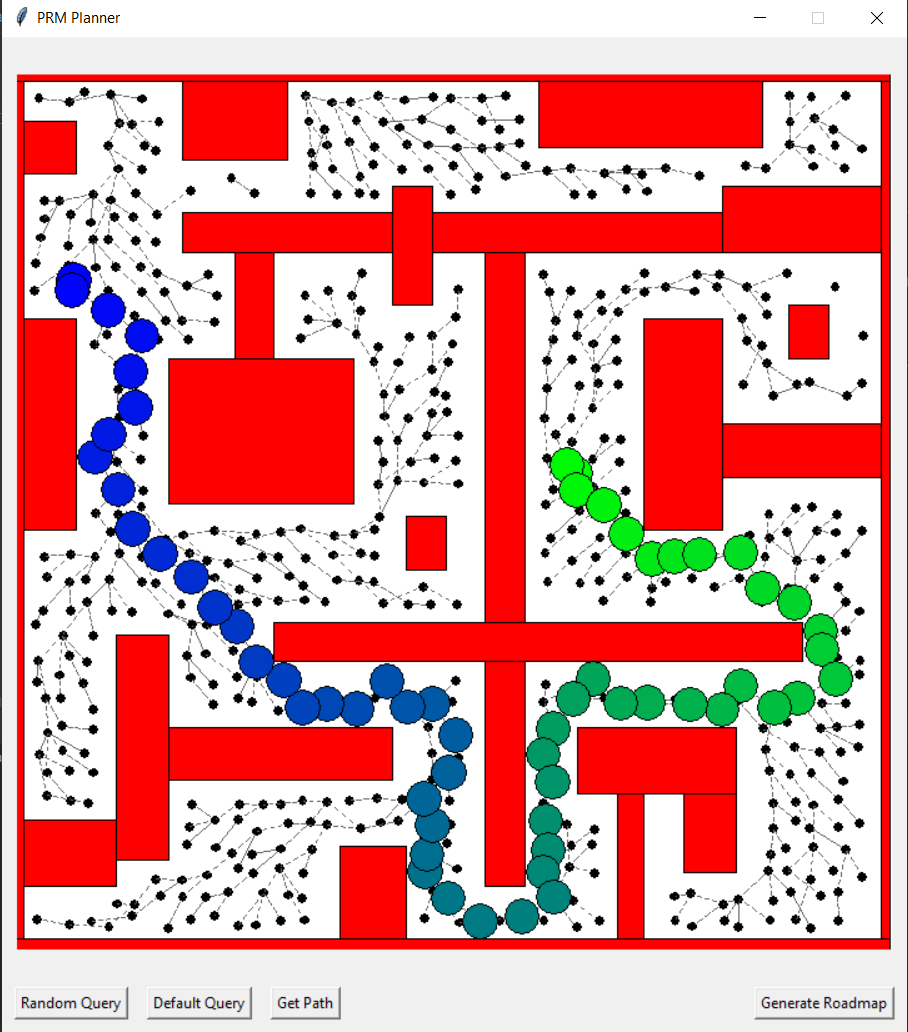


Figure: Result using uniform sampling with noise addition and default query.

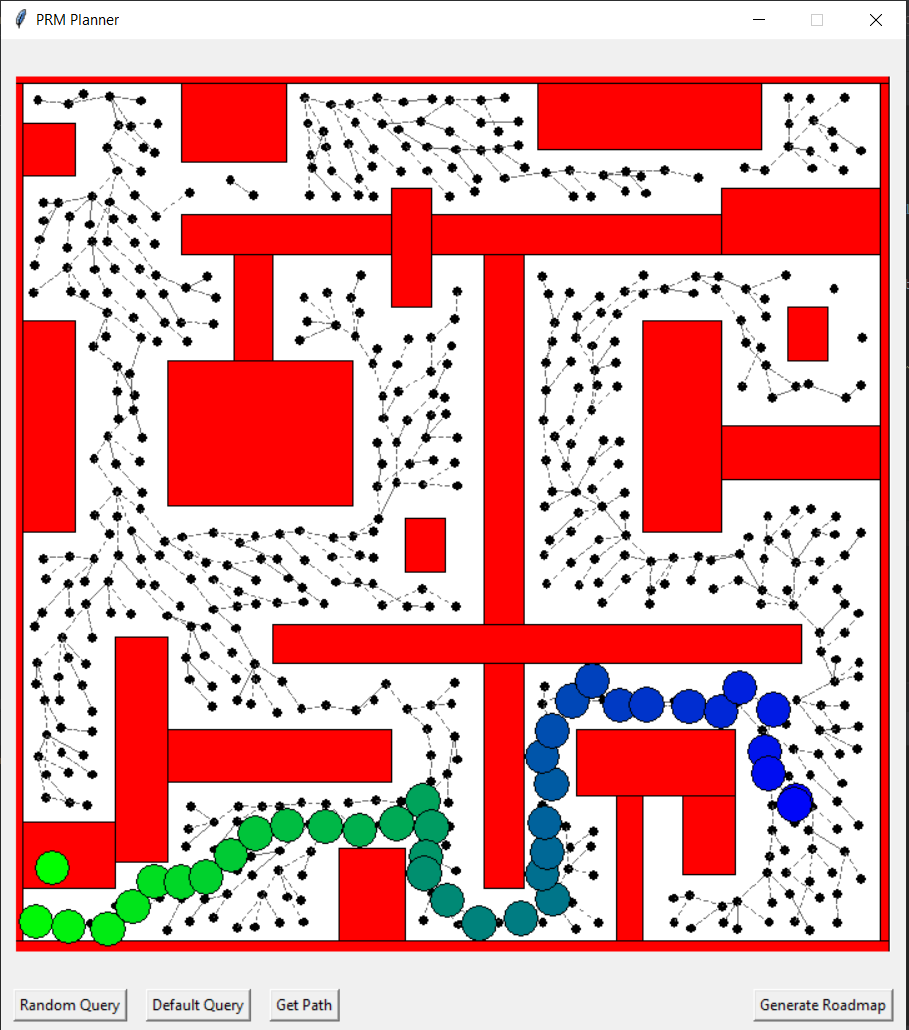


Figure: Result using uniform sampling with noise addition and random query.

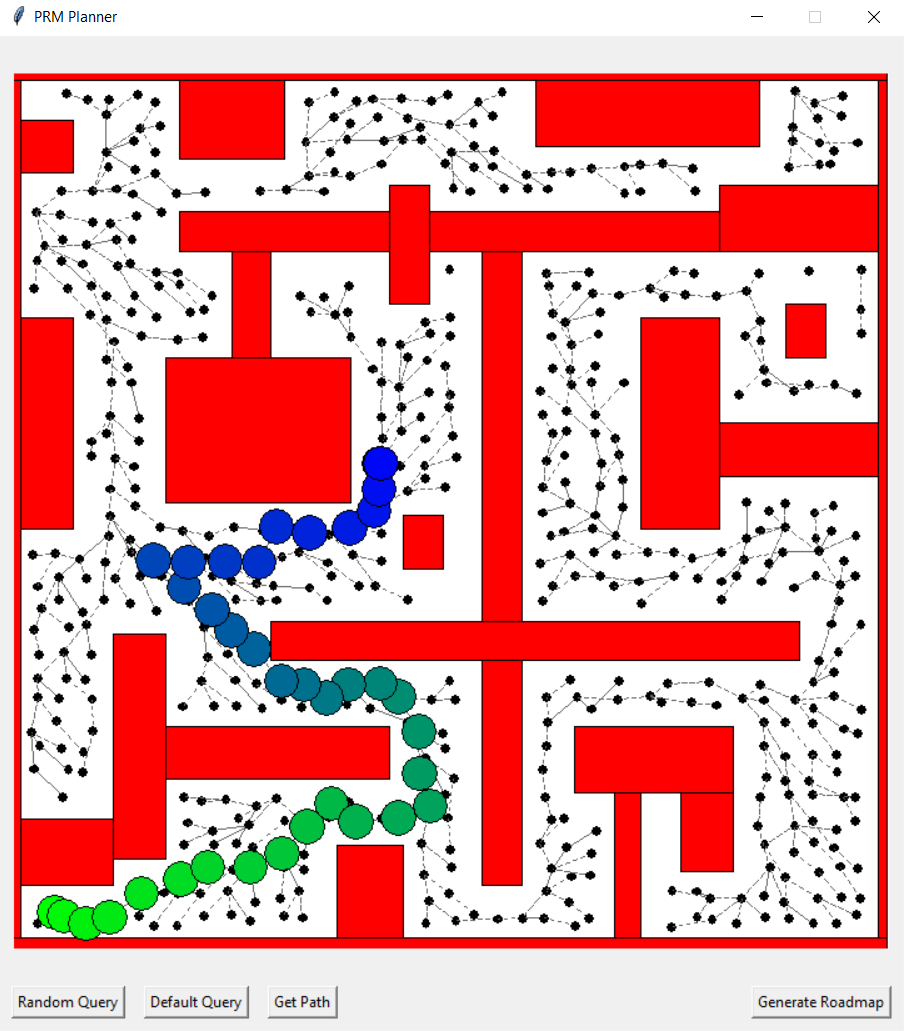


Figure: Result using uniform sampling with noise addition and random query.

We got best result using uniform sampling with noise addition and BFS local planner. BFS gives the connectivity better than DFS. This works well with any random query where the start or goal points are on the obstacles or in the free space, astar will give you’re the path as accurate as possible.

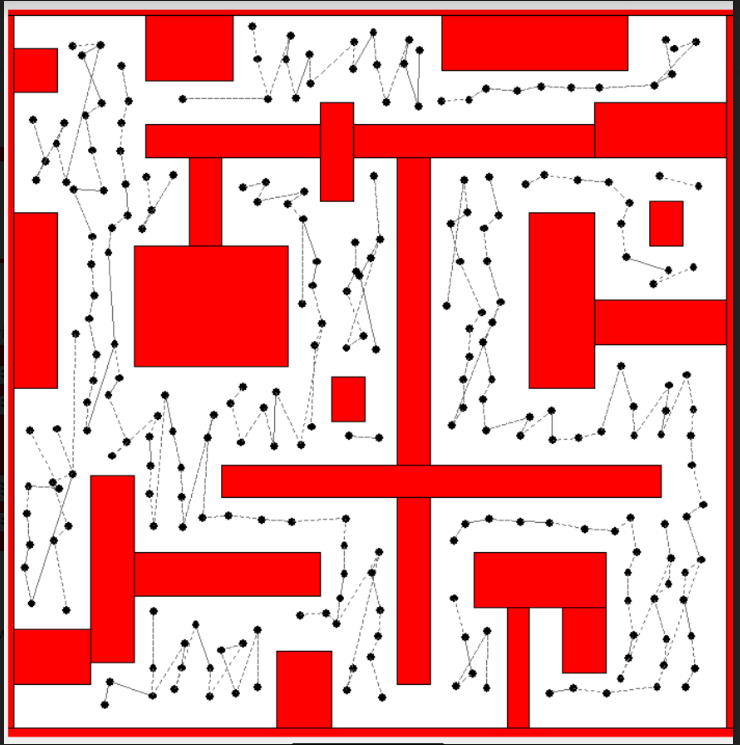


Figure: Result using grid-based approach.

We used the grid and mapped throughout the image to generate random noise in that portion. Because, the grid moves stepwise, some of the vertices remain unconnected. Although this technique can be used to generate edges where the path is narrow for example, top right corner. (DFS local planner)

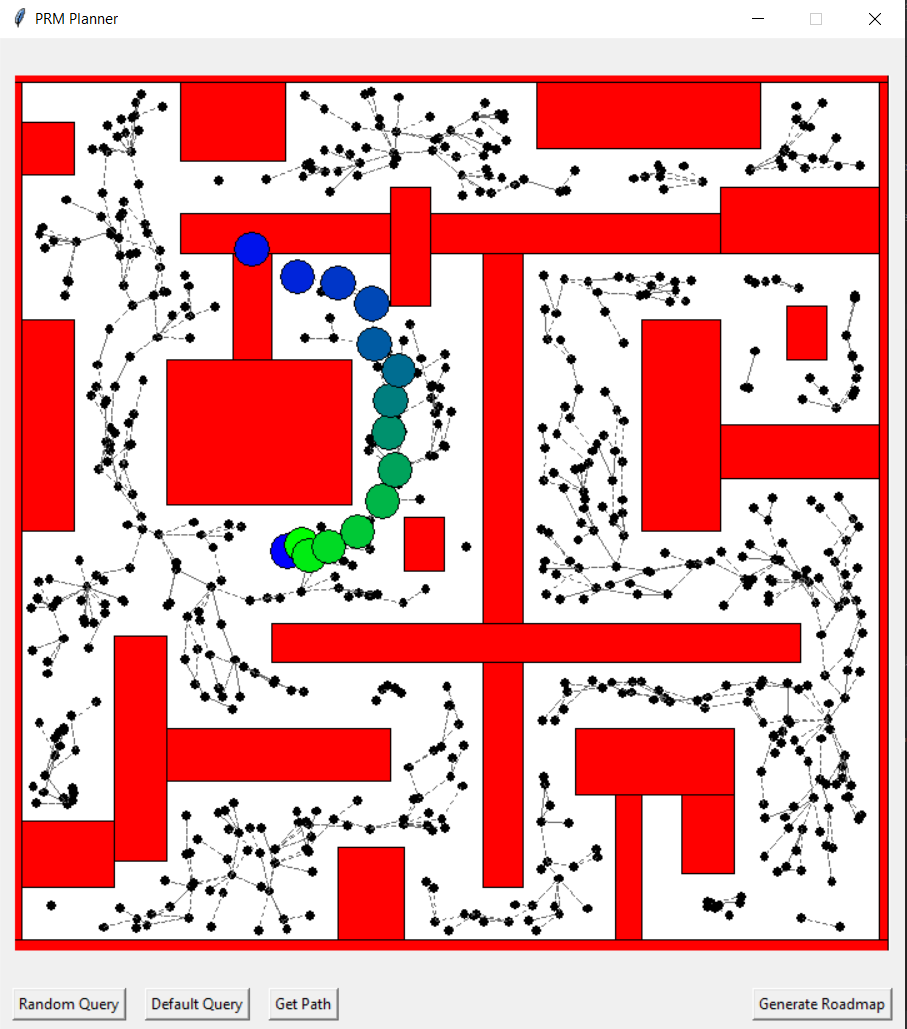


Figure: Result using random sampling and random query.

We used random uniform and because of randomness some points are too closed to each other and we can see the clusters of the points. At some location vertices are not connected and astar couldn’t find path.